I. In the claims:

1. (Currently Amended) In a multiprocessor computer system having multiple interconnected processing nodes each with one or more processors and physical memory, a data structure for storing execution history data indicating of states of threads that are used for providing mutual exclusion between current and next generation data elements, comprising:

a first level bit mask stored in physical memory accessible to all nodes and containing a bit per node, the bit indicating whether the corresponding nodes contains a processor that has not yet passed through a quiescent state; and

a second level bit mask stored in the physical memory of each processing node and containing a bit per processor associated with a particular node identified in the first level bit mask, the bit indicating whether the corresponding processor has not yet passed through a quiescent state.

2. (Currently Amended) In a multiprocessor computer system having multiple interconnected processing nodes each with one or more processors and physical memory, a data structure stored in the physical memory on each node for storing a number of the current generation of data elements being processed by a processor on a node, the data structure comprising a variable stored in the physical memory of each node and containing the current generation number, wherein the current generation number on the nodes are updated in lockstep so that the nodes have local access to copies of the current generation number.

3. (Currently Amended) In a multiprocessor computer system having multiple interconnected processing nodes each with one or more processors and physical memory, a method for a processor to maintain a summary of thread activity as part of a method for providing mutual exclusion between current and next generation data elements, comprising:

determining from a first data structure on the processor's node if the processor has passed through a quiescent state;

if so, determining from a second data structure on the processor's node if all other processors on its node have passed through a quiescent state; and

if so, indicating in a third data structure accessible to all nodes that all processors on the processor's node have passed through a quiescent state.

4. (Currently Amended) The method of claim 3 wherein if the processor determines from the first data structure on the processor's node that the processor has not passed through a quiescent state, having a callback processor check if the processor has passed through a quiescent state and, if so, having the processor indicate in the first data structure that it has passed through a quiescent state.

5. (Currently Amended) The method of claim 3 wherein if the processor determines from the third data structure accessible to all nodes that the processor is the last processor to pass through a quiescent state, having the processor update a fourth data structure stored in the memory of each node for storing a number of the current generation of elements being processed on the node.

6. (Currently Amended) The method of claim 3 wherein if the processor determines from the third data structure accessible to all nodes that it is the last processor to pass through a quiescent state, having a callback processor determine if there are callbacks waiting for a subsequent generation, and, if so, updating the first data structure on each node and the third data structure accessible to all nodes to indicate that all processors need to pass through an additional quiescent state.

7. (New) The method of claim 3, wherein the first, second, and third data structures are the same data structure.

8. (New) The method of claim 3, wherein the first and second data structures are the same data structure.

9. (New) The method of claim 3, wherein the second and third data structures are the same data structure.

10. (New) The method of claim 3, wherein the first and third data structures are the same data structure.